

日 本 国 特 許 庁

PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年10月25日

出 願 番 号

Application Number:

特願2000-332110

出 願 人

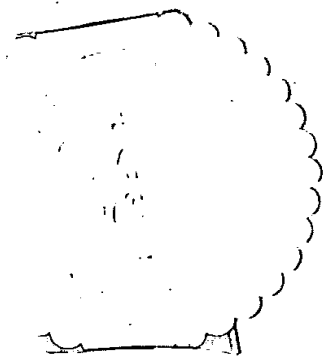
Applicant(s):

株式会社日立製作所

日立ソフトウェアエンジニアリング株式会社

09/810,191
Mitsumori et al

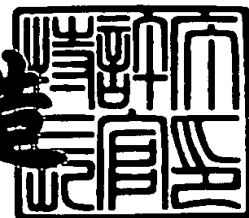
phone 703-684-1120
ASA-990



2001年 3月16日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2001-3020099

【書類名】 特許願

【整理番号】 K00016471

【提出日】 平成12年10月25日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/45

【請求項の数】 10

【発明者】

【住所又は居所】 神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所 ソフトウェア開発本部内

【氏名】 三森 征人

【発明者】

【住所又は居所】 神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所 ソフトウェア開発本部内

【氏名】 浅香 真司

【発明者】

【住所又は居所】 神奈川県横浜市中区尾上町6丁目81番地 日立ソフトウェアエンジニアリング株式会社内

【氏名】 細谷 洋行

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社日立製作所

【特許出願人】

【識別番号】 000233055

【氏名又は名称】 日立ソフトウェアエンジニアリング株式会社

【代理人】

【識別番号】 100075096

【弁理士】

【氏名又は名称】 作田 康夫

【先の出願に基づく優先権主張】

【出願番号】 特願2000-200068

【出願日】 平成12年 6月28日

【手数料の表示】

【予納台帳番号】 013088

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9902691

【包括委任状番号】 9102708

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 コンパイル方法および記録媒体

【特許請求の範囲】

【特許請求の範囲】

【請求項 1】

ソースプログラムから目的プログラムを生成するコンパイラにおいて、
コンパイルした目的プログラムを目的プログラムファイルに格納し、
前記目的プログラムファイルに該目的プログラムのソースプログラムを該目的プログラムに対応付けて格納すること
を特徴とするコンパイル方法。

【請求項 2】

請求項 1 において、目的プログラムを生成する前に、
コンパイラに入力するソースプログラムと前記目的プログラムファイルに格納されている該目的プログラムに対応するソースプログラムを比較し、
以前コンパイルした時とのソースプログラムに対する変更部分を検出し、
入力したソースプログラムの変更部分をコンパイルし、
コンパイルした目的プログラムと前記入力ソースプログラムの変更部分を前記目的プログラムファイルに格納すること
を特徴とするコンパイル方法。

【請求項 3】

ソースプログラムから目的プログラムを生成するコンパイラにおいて、
コンパイラに入力するソースプログラムの構文を解析してソース情報を取得し、
前記ソース情報をコンパイルした目的プログラムを目的プログラムファイルに格納し、
前記目的プログラムファイルに該目的プログラムのソース情報を該目的プログラムに対応付けて格納すること
を特徴とするコンパイル方法。

【請求項 4】

請求項 3 において、目的プログラムを生成する際に、

前記コンパイラに入力するソースプログラムの構文を解析したソース情報と前記目的プログラムファイルに格納されているソース情報を比較し、
以前コンパイルした時とのソース情報に対する変更部分を検出し、
入力したソースプログラムのソース情報の変更部分をコンパイルし、
目的プログラムと前記ソース情報の変更部分を前記目的プログラムファイルに格納する

ことを特徴とするコンパイル方法。

【請求項 5】

請求項 2 または請求項 4 において、
前記ソースプログラムまたは前記ソース情報の比較は、ソースプログラムの手続き単位をおこない、
前記ソースプログラムまたは前記ソース情報の変更部をコンパイルし、
前記目的プログラムファイルに手続き単位に目的プログラムとソースプログラムまたはソース情報を対応付けて格納する
ことを特徴とするコンパイル方法。

【請求項 6】

請求項 2 または請求項 4 において、
前記ソースプログラムまたは前記ソース情報の比較は、ソースプログラムの処理ステップ単位をおこない、
前記ソースプログラムまたは前記ソース情報の変更部をコンパイルし、
前記目的プログラムファイルに処理ステップ単位に目的プログラムとソースプログラムまたはソース情報を格納する
ことを特徴とするコンパイル方法。

【請求項 7】

請求項 2 または請求項 4 において、
コンパイラに入力するソースプログラムと目的プログラムファイルに格納されている該目的プログラムに対応するソースプログラムまたはソース情報の比較するか否かの指定を入力し、
比較しない指定が入力された場合は、コンパイラに入力したソースプログラムの

すべてを変更部分としてコンパイルすること
ことを特徴とするコンパイル方法。

【請求項 8】

請求項 1 または請求項 3 において、
目的プログラムファイルに目的プログラムのソースプログラムまたはソース情報を格納するか否かの指定を入力し、
格納しない指定が入力された場合は、目的プログラムを目的プログラムファイルに格納する
ことを特徴とするコンパイル方法。

【請求項 9】

請求項 1 から請求項 8 のいずれかのコンパイル方法を実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項 10】

請求項 1 から請求項 8 のいずれかのコンパイル方法により生成された目的プログラムファイルを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、ソースプログラムを入力してコンパイルし、作成した目的プログラムを目的プログラムファイルに格納するコンパイル方法に関する。

【0002】

【従来の技術】

従来のコンパイラは、ファイル単位でソースプログラムファイルを入力し、ソースプログラムをコンパイルしていた。このため、複数の手続きが記述された 1 つのソースプログラムをコンパイルし、その後、そのソースプログラム内の 1 つの手続きのみを変更して、再び同じソースプログラムをコンパイルすると、同じソースプログラム内の変更していない手続きを含めた全ての手続きについてコンパイルしている。

【 0 0 0 3 】

上記のコンパイル手続きを自動的におこなうツールとしてmakeツールがしばしば用いられている。makeツールは、複数のソースプログラムから目的プログラムを生成するためのツールで、ソースプログラムファイルの日付を管理して、目的プログラムファイルより新しい日付のソースプログラムファイルのみをコンパイルするものである。

【 0 0 0 4 】

【発明が解決しようとする課題】

上記の従来技術は、変更があるかないかを区別するのにソースプログラムファイルの日付を参照していた。コンパイラはファイル単位でソースプログラムを入力する為、複数の手続きが記述された1つのソースプログラムファイル内で、他の手続きには全く影響しない手続きが1つのみ変更されたり、日付のみ変更された場合、コンパイラはソースプログラムファイル内全ての手続きについてコンパイルしてしまう。その為、わずかな変更やファイルの日付のみを変更した場合でも、無駄なコンパイル処理を行い、コンパイル時間がかかるという問題がある。

【 0 0 0 5 】

つまり、従来の技術では、複数の手続きが記述されたソースプログラム内の1つの手続きのみの変更でも、コンパイラは全ての手続きについてコンパイルする。その為、わずかな変更でもコンパイルする時間は短縮されず、無駄なコンパイル処理を行うという問題がある。

【 0 0 0 6 】

本発明の目的は、入力されたソースプログラムファイルが以前に既にコンパイルされ、コンパイルされた時からソースプログラム中の手続きが変更されていない場合は、その手続きのコンパイル処理を省き、コンパイル時間の短縮を図ることである。

【 0 0 0 7 】

【課題を解決するための手段】

上記目的を達成するために、本発明によるコンパイラは、コンパイルするソースプログラムを識別する手段と、変更のあるソースプログラム内の手続きについ

てのみコンパイルする手段、ソースプログラムを目的プログラムと一緒に目的プログラムファイル内に格納する手段を備える。

【0008】

上記構成により、まず、コンパイラは入力されたソースプログラムと、以前にコンパイルされ目的プログラムファイル内に格納されている目的プログラムに対応するソースプログラムを、ソースプログラム内の手続きごとに比較し、変更のあった手続きを識別する。つぎに、変更のあった手続きのみコンパイルをおこなない目的プログラムを得る。さらに、コンパイラした目的プログラムとコンパイルしたソースプログラムを対応付けて目的プログラムファイル内に格納する。

【0009】

また、上記目的を達成するために、コンパイラはソースプログラムファイルを入力してソースプログラムをコンパイルする際、コンパイルしたソースプログラムを識別する為に、ソースプログラムの構文を解析した情報をソース情報として、目的プログラムと一緒に目的プログラムファイル内に格納する。

【0010】

利用者が再度同じソースプログラムをコンパイルする場合、コンパイラはコンパイルしている手続きの構文を解析したあとで、目的プログラムファイル内に格納されたソース情報を読み込み、コンパイラに入力されるソース情報と比較する。比較した結果、一致する場合はその手続きのソースプログラムは変更が無いと判断し、以後のコンパイル処理は行わない。比較した結果、一致しない場合はその手続きのソースプログラムが変更されたと判断し、変更された手続きをコンパイルして目的プログラムを作成する。その後、目的プログラムファイル内の変更前の手続きの目的プログラムを、今回コンパイルして作成した目的プログラムに更新し、目的プログラムファイル内の変更前の手続きのソース情報も、コンパイラに入力され構文を解析した情報をソース情報としたものに更新する。

【0011】

【発明の実施の形態】

以下、図面を用いて本発明の実施の形態を説明する。

【 0 0 1 2 】

(実施例 1)

図 1 は本発明を用いたコンパイラの構成図である。1 0 1 は記憶装置上に格納されているソースプログラムファイル、1 0 2 は本発明を実施するコンパイラ本体、1 0 3 はソースプログラムファイルを入力する処理、1 0 4 は入力されたソースプログラムの構文を解析する処理、1 0 5 は構文を解析した情報と目的プログラムファイル内のソース情報を比較し、以後の必要なコンパイル処理を行う必要があるかを判断する処理で、その中の1 0 6 では目的プログラムファイル内のソース情報を読取り、1 0 7 では読取ったソース情報と現在コンパイルしている手続きの構文を解析した情報を比較する。1 0 8 は一連のコンパイル処理において一番複雑で時間のかかる最適化を行う処理、1 0 9 はコンパイルして作成した目的プログラムや構文を解析した情報を、目的プログラムファイル内に格納する処理で、その中の1 1 0 では目的プログラムを目的プログラムファイルに格納、又は既に目的プログラムファイル内に同じ手続きの目的プログラムが存在する場合は、その目的プログラムを最新の目的プログラムに更新し、1 1 1 ではコンパイルした手続きの構文を解析した情報をソース情報として目的プログラムファイルに格納、又は既に目的プログラムファイル内に同じ手続きのソース情報が存在する場合は、そのソース情報を最新のソース情報に更新する。1 1 2 はコンパイルして記憶装置上に格納される目的プログラムファイルである。

【 0 0 1 3 】

図 2 は、1 0 5 のソース情報解析部のフローチャートである。2 0 1 から2 0 3 が1 0 6 のソース情報読取り部の処理で、2 0 4 から2 0 5 が1 0 7 のソース情報比較部の処理である。

【 0 0 1 4 】

2 0 1 では、コンパイルしているソースプログラムファイルに目的プログラムファイルが存在するかを判断し、目的プログラムファイルが存在しない場合は、コンパイルしているソースプログラムの全ての手続きについて、1 0 8 の最適化部から1 0 9 の目的プログラムファイル作成部の処理を行う。目的プログラムファイルが存在する場合は2 0 2 の処理を行う。

【 0 0 1 5 】

2 0 2 では、コンパイルしているソースプログラムファイルの目的プログラムファイル内に、以前格納したソース情報が存在するかどうかを判断し、目的プログラムファイル内にソース情報が存在しない場合は、コンパイルしているソースプログラム内の全ての手続きについて、1 0 8 の最適化部から1 0 9 の目的プログラムファイル作成部の処理を行う。目的プログラムファイル内にソース情報が存在する場合は2 0 3 の処理を行う。

【 0 0 1 6 】

2 0 3 では、コンパイルしているソースプログラムファイルの目的プログラムファイル内からソース情報を読み込む処理を行う。

【 0 0 1 7 】

2 0 4 では、2 0 3 で読み込んだソース情報の中に現在コンパイルしている手続きのソース情報が存在するかどうかを判断する。もし、手続きのソース情報が存在しない場合は、その手続きが新規に追加されたと判断し、その新規の手続きについて1 0 8 の最適化部から1 0 9 の目的プログラムファイル作成部の処理を行う。2 0 3 で読み込んだソース情報の中に現在コンパイルしている手続きのソース情報が存在する場合は2 0 5 の処理を行う。

【 0 0 1 8 】

2 0 5 では、現在コンパイルしている手続きの構文を解析した情報と、目的プログラムファイル内に存在している該当する手続きのソース情報を比較し、一致する場合はその手続きのソースプログラムは変更が無いと判断し、1 0 8 の最適化部から1 0 9 の目的プログラムファイル作成部の処理は行わない。構文を解析した情報とソース情報が一致しない場合は、その手続きのソースプログラムが変更されたと判断し、その手続きについて1 0 8 の最適化部から1 0 9 の目的プログラムファイル作成部の処理を行う。

【 0 0 1 9 】

この2 0 1 から2 0 5 までの処理によって、現在コンパイルしているソースプログラムの変更部分を手続き単位に検出することができる。

【 0 0 2 0 】

図 3 は、 1 0 9 の目的プログラムファイル作成・更新部のフローチャートである。

【 0 0 2 1 】

3 0 1 では、コンパイルしているソースプログラムファイルの目的プログラムファイルが存在するかを判断する。もし、目的プログラムファイルが存在しない場合は、コンパイルしているソースプログラム内の全ての手続きについて 3 0 5 の処理を行う。

【 0 0 2 2 】

3 0 2 では、コンパイルしているソースプログラムファイルの目的プログラムファイル内に、以前格納したソース情報が存在するかどうかを判断する。もし、目的プログラムファイル内にソース情報が存在しない場合は、コンパイルしているソースプログラム内の全ての手続きについて 3 0 5 の処理を行う。

【 0 0 2 3 】

3 0 3 では、コンパイルしているソースプログラムの目的プログラムファイル内に既にソース情報が存在し、そのソース情報の中に、現在コンパイルしている手続きのソース情報が存在するかどうかを判断する。もし、コンパイルしている手続きのソース情報が目的プログラムファイル内に存在しない場合は、コンパイルしている手続きについて 3 0 5 の処理を行う。

【 0 0 2 4 】

3 0 4 では、目的プログラムファイル内に既に存在するコンパイルした手続きの目的プログラムとソース情報を、今回コンパイルして作成した目的プログラムと構文を解析した情報をソース情報としたものに更新する。

【 0 0 2 5 】

3 0 5 では、該当する手続きをコンパイルして作成した目的プログラムと構文を解析した情報をソース情報として、目的プログラムファイル内に新規に格納する。

【 0 0 2 6 】

図 4 は、利用者からコンパイラへ本発明の処理を行うかどうかの指示があった

場合のフローチャートである。

【 0 0 2 7 】

4 0 1 では、現在コンパイルしているソースプログラムの構文を解析した情報と、目的プログラムファイル内のソース情報を比較するかどうかを判断し、利用者から比較する処理を行わない指示があった場合は、コンパイルしているソースプログラムの全ての手続きについて、1 0 8 の最適化部から1 0 9 の目的プログラムファイル作成部の処理を行う。

【 0 0 2 8 】

4 0 2 では、コンパイルした手続きの構文を解析した情報を、ソース情報として目的プログラムファイル内に格納するかどうかを判断し、利用者から目的プログラムファイル内にソース情報を格納しない指示があった場合は、1 1 1 のソース情報作成・更新部の処理は行わず、目的プログラムファイル内には目的プログラムのみが格納される。

【 0 0 2 9 】

図 5 は、ソース情報を格納した場合の目的プログラムファイルの構造である。1 1 1 は記憶装置上に格納された目的プログラムファイルで、その中に各手続きの目的プログラムが格納された目的プログラム部 5 0 1 がある。目的プログラムファイル内にはコンパイルした各手続きの構文を解析した情報もソース情報としてソース情報部 5 0 2 に格納される。目的プログラムとソース情報が同じ目的プログラムファイル内に格納されているので、利用者は今までと同じコンパイラの使用方法で本発明を利用することが出来る。

【 0 0 3 0 】

(実施例 2)

図 6 に本発明を用いた他の実施例のコンパイラの構成図をしめす。記憶装置上に格納されているソースプログラムファイル 1 0 1 を、本発明によるコンパイラ 1 0 2 が入力してコンパイルし、記憶装置に目的プログラムファイル 1 1 2 を出力する。この目的プログラムファイル 1 1 2 の構造は、先の実施例と同様にコンパイル結果である目的プログラムと該目的プログラムのソースプログラムから構成され、詳細は後述する。

【 0 0 3 1 】

以下、本発明のコンパイラ 1 0 2 の構成を詳細に説明する。コンパイラ 1 0 2 は、ソースプログラム入力部 1 0 3 と構文解析部 1 0 4 とソースプログラム解析部 1 0 5 と目的プログラムファイル作成部 1 0 9 から構成される。ソースプログラム入力部 1 0 3 は、ソースプログラムファイル 1 0 1 からコンパイル対象のソースプログラムを入力し、構文解析部 1 0 4 は、入力したソースプログラムの構文を解析し、言語仕様を満足しているかを判別する。ソースプログラム解析部 1 0 5 は、入力したソースプログラムの変更部分を検出して、ソースプログラムのコンパイルを行うか否かを手続き単位に判定するものであり、目的プログラムファイル認識部 6 0 1、ソースプログラム読み取り部 6 0 2、ソースプログラム比較部 6 0 3、最適化処理部振り分け部 6 0 4 からなる。目的プログラムファイル作成部 1 0 9 は、手続き単位にコンパイルをおこない、目的プログラムファイル 1 1 2 内にコンパイル結果である目的プログラムと、目的プログラムに対応するソースプログラムを出力するものであり、目的プログラム生成部 6 0 5、目的プログラム出力部 6 0 6、ソースプログラム出力部 6 0 7 からなる。

【 0 0 3 2 】

目的プログラムファイル認識部 6 0 1 は、コンパイルするように指定されたソースプログラムファイル 1 0 1 に対応するソースプログラムが目的プログラムファイル 1 1 2 にすでに存在するか確認を行う。ソースプログラム読み取り部 6 0 2 は、目的プログラムファイル 1 1 2 内のソースプログラムを手続きごとに読み取る。ソースプログラム比較部 6 0 3 は、読み取った手続きのソースプログラムとコンパイラに指定されたソースプログラムを手続きごとに比較し、ソースプログラムファイル 1 0 1 の変更部分を検出する。最適化処理部振り分け部 6 0 4 は、コンパイルする手続きのソースプログラムに変更部分が検出された場合、つぎの最適化処理部 1 0 8 を行うよう指示する。最適化処理部 1 0 8 は、コンパイルしたソースプログラムを実行する時の速度を速くする為、ソースプログラムを解析し、意味の無い処理ステップを削除したり、並べ替えたりする処理を行う。また、指定によっては、生成する目的プログラムのサイズが小さくなるような処理を行う。最適化を行うのレベルをコンパイル時に指定でき、レベルによりコンパ

イル時間も長くなる。

【0033】

目的プログラム生成部605は、最適化処理をおこなったソースプログラムをコンパイルし目的プログラムを生成する。目的プログラム出力部606は、コンパイルして生成された目的プログラムを目的プログラムファイル103内に出力し、ソースプログラム出力部607は、コンパイルしたソースプログラムを手続き単位に目的プログラムファイル103内に出力する。

【0034】

図7に、ソースプログラム解析部105の処理フローをしめす。最初にコンパイラに指定されたソースプログラムファイル101に対応する目的プログラムファイル112が存在するか判別する(201ステップ)。判別の結果、目的プログラムファイル112が存在しないときは、最初にコンパイルする場合であり、ソースプログラムファイル101内の全ての手続きのソースプログラムをコンパイルするため、全ての手続きについて後述のステップ701の処理を行う。目的プログラムファイル112が存在する場合には、以前コンパイルした時に格納したソースプログラムが、目的プログラムファイル112内に存在するかどうかを判別する(202ステップ)。判別の結果、目的プログラムファイル112内にソースプログラムが存在しない場合は、ソースプログラムファイル101内の全ての手続きのソースプログラムをコンパイルする為、全ての手続きについて後述のステップ701の処理を行う。目的プログラムファイル112内にソースプログラムが存在する場合には、目的プログラムファイル112内に、コンパイルしている手続きのソースプログラムが存在するかどうかを判別する(204ステップ)。この判別の結果、コンパイルしている手続きのソースプログラムが存在しない場合は、その手続きが新規に追加されたと判断し、その手続きをコンパイルするため、その手続きについて後述のステップ701の処理を行う。コンパイルしている手続きのソースプログラムが存在する場合には、コンパイルしているソースプログラムファイル101に対応するソースプログラムを目的プログラムファイル103内から、手続き単位に読み込む(203ステップ)。つぎに、コンパイルする手続きのソースプログラムと、目的プログラムファイル112内の

対応する手続きのソースプログラムを比較し、一致しているかを判別する（205ステップ）。この判別の結果、コンパイルしている手続きのソースプログラムと、目的プログラムファイル112内の対応する手続きのソースプログラムが一致しない場合は、その手続きが変更されたと判断し、その手続きをコンパイルするため、その手続きについて次のステップ701の処理を行う。一致する場合は、その手続きは変更されていないため、コンパイルは必要ないので次のステップ701の処理は行わない。なお、ソースプログラム中で複数の手続きが変更されていた場合には、ステップ203、ステップ204の処理を複数回おこなうようにする。ステップ701では、変更のあったソースプログラムの手続きや、ソースプログラムについて最適化処理をおこなうようにする。上記の処理フローにより、コンパイルしている手続きのソースプログラムの変更部分を検出し、手続きごとにコンパイルを行うかを振り分けることができる。

【0035】

図8は、目的プログラムファイル作成部108の処理を説明するフローチャートである。最初に、先に説明したソースプログラム解析部105で、ソースプログラムの手続きの変更を検出し、手続きごとに最適化処理されたかを判定する（801ステップ）。判定の結果、最適化処理された手続きがある場合には、つぎのステップ802、ステップ803、ステップ804の処理をおこなう。

【0036】

ステップ802では、ソースプログラムの手続きをコンパイルして目的プログラムを生成し、ステップ803では、コンパイルした手続きの目的プログラムを目的プログラムファイル112内に出力し、既にコンパイルした手続きの目的プログラムが目的プログラムファイル112内に存在する場合は、その目的プログラムを更新する。ステップ804は、コンパイルした手続きのソースプログラムを目的プログラムファイル112内に出力し、既にコンパイルした手続きのソースプログラムが目的プログラムファイル112内に存在する場合は、そのソースプログラムを更新する。このようにして、ソースプログラムファイル101の変更された手続きだけをコンパイルし、生成された目的プログラムを目的プログラムファイル112に格納するとともに、対応するソースプログラムも、目的プロ

グラムに関連付けて目的プログラムファイル 1 1 2 に格納するようにする。

【 0 0 3 7 】

上記の実施例では、初めてコンパイルをおこなう場合でもコンパイル時に指定されたソースプログラムファイル 1 0 1 が目的プログラムファイル 1 1 2 に存在するか判定をおこなっているが、このように明らかに目的プログラムファイル 1 1 2 にソースプログラムがない場合には、図 9 のような処理フローを追加してもよい。つまり、コンパイラがソースプログラムを比較するか否かの指示を判別するようにし（9 0 1 ステップ）、比較する処理を行わない指示があった場合は、ソースプログラム解析部 1 0 5 の処理は行わずに、すべての手続きについて、最適化処理とコンパイルをおこなうようにする。これにより、目的プログラムファイル 1 1 2 内にソースプログラムが存在していないということが既にわかっている場合は、ソースプログラム解析部 1 0 5 の処理を省略でき、コンパイル時間が短縮できる。

【 0 0 3 8 】

また、目的プログラムファイル内には、手続きごとに目的プログラムと対応するソースプログラムが格納されているが、ソースプログラムが不要の場合、例えば、他者にソースプログラムを開示したくない場合や今後再びコンパイルする必要がない場合に、図 1 0 のような処理フローを追加してもよい。つまり、コンパイラが目的プログラムファイル内にソースプログラムを格納するか否かの指示を判別するようにし（1 0 0 1 ステップ）、ソースプログラムを格納する指示があった場合は、先に述べたように目的プログラムに対応するソースプログラムを目的プログラムファイル 1 1 2 に出力するようにする。これにより不要なソースプログラムが目的プログラムファイルの格納されることを防止できる。

【 0 0 3 9 】

上記実施例では、手続き単位にソースプログラムの変更部を検出し、変更部を含む手続きをコンパイルし、コンパイルした目的プログラムと該目的プログラムのソースプログラムを手続きごとに目的プログラムファイルに格納するようにしている。ソースプログラムを変更する場合に、定数の変更や参照する変数の変更がよくおこなわれる。このような変更の場合は、目的プログラムのサイズは変わ

らずに一部のコードが変更されることがおおい。このため、変更部の検出をソースプログラムのステップ単位におこない、ステップ単位にコンパイルをおこない、目的プログラムと該目的プログラムのソースプログラムをステップ単位に目的プログラムファイルに格納するようにしてもよい。この例では、コンパイル時間の短縮の効果より大きくなる。

【 0 0 4 0 】

つぎに本発明のコンパイラが出力する目的プログラムファイルについて詳細に説明する。図 1 1 にしめすように、記憶装置上に格納された目的プログラムファイル 1 1 2 の中には、各手続きの目的プログラムが更新しやすいように別々に格納された目的プログラム 1 1 0 1 と、コンパイルしたソースプログラムの全ての手続きを更新しやすいように別々に格納したソースプログラム 1 1 0 2 がある。このように、目的プログラムとコンパイルしたソースプログラムが同じ目的プログラムファイル 1 0 3 内に保持されているので、利用者は操作性を変える必要はない。また、ソースプログラムは、同じ目的プログラムファイル 1 1 2 内に保持しなくても、手続きの目的プログラム 1 1 0 1 と手続きのソースプログラム 1 1 0 2 が記憶装置上で、対応付けられて格納されていれば本発明は実現可能である。

【 0 0 4 1 】

図 1 2 は、ソースプログラムファイル 1 0 1 に格納されているソースプログラムの一例をしめす図である。ソースプログラムには、`func1` と `func2` の 2 つの手続きが存在し、本発明のコンパイラにより図 1 3 にしめす目的プログラムファイル 1 1 2 が生成される。目的プログラムファイルは、目的プログラム部とソースプログラム部から構成され、目的プログラム部には、`func1` の目的プログラム 1 3 0 1 と `func2` の目的プログラム 1 3 0 2 が生成される。ソースプログラム部には、コンパイル時に目的プログラムに対応して、`func1` のソースプログラム 1 3 0 3 と `func2` のソースプログラム 1 3 0 4 が格納される。

【 0 0 4 2 】

ソースプログラムファイル 1 0 1 の `func1` のみ修正して再度コンパイルした場

合について説明する。ソースプログラム解析部 1 0 5 で、ソースプログラムファイル 1 0 1 と目的プログラムファイル 1 1 2 のソースプログラム部の 1 3 0 3 が比較され、func1 の変更を検出する。つぎに目的プログラムファイル部 1 0 9 で、func1 のみコンパイルを行い、目的プログラムファイル 1 1 2 内の func1 の目的プログラム 1 3 0 1 と func1 のソースプログラム 1 3 0 3 を更新する。

【 0 0 4 3 】

つぎに図 1 4 を用いて、本発明のコンパイラが出力する目的プログラムファイル 1 1 2 の構成を説明する。目的プログラムファイル 1 1 2 は、コンパイラが出力する目的プログラムとソースプログラム部とヘッダー部から構成される。ヘッダー部には、コンパイルした各手続きごとに目的プログラムとソースプログラムが関連付けられるように、それぞれの格納場所が格納される。

【 0 0 4 4 】

さらに、ヘッダー部には、該手続きをコンパイルした時のコンパイラのバージョンやコンパイルした時の最適化レベルも格納されている。使用するコンパイラが新しくなったり、コンパイル時の最適化レベルが変わった場合は、各手続きの目的プログラムの作成条件を合わせる為に全ての手続きがコンパイル対象となるが、ソースプログラムを比較するだけでは検出できないので、コンパイルした時のコンパイラのバージョンやコンパイルした時の最適化レベルも一緒に比較する。それにより、コンパイルするコンパイラやコンパイルした時の最適化レベルが変わっても、不正にコンパイルを省略することがなくなる。

【 0 0 4 5 】

【発明の効果】

本発明によれば、コンパイラに指定されたソースプログラムファイルをコンパイルせずに、ソースプログラムの変更部分をコンパイルすることができるので、コンパイル時間を短縮でき、しかもコンパイルしたソースプログラムが目的プログラムファイル中にあるため、利用者は操作性を変えことなく、コンパイル時間の短縮を図ることができる。特に、複数の手続きが記述されたソースプログラムであって、手続き単位にリコンパイルをおこなうことにより、コンパイル時の最適化処理に膨大な時間がかかる手続きが変更されていない場合は、コンパイル

時間を大幅に短縮できる。

【 0 0 4 6 】

また、ソースプログラムファイルを誤って削除した場合でも、目的プログラムファイル内からソースプログラムが取り出せる為、削除前の環境に復旧させることができる。

【図面の簡単な説明】

【図 1】

本発明のコンパイラの実施例 1 の構成図。

【図 2】

実施例 1 のソース情報解析部のフローチャート。

【図 3】

実施例 1 のソース情報作成・更新部のフローチャート。

【図 4】

実施例 1 の利用者からコンパイラへの本発明を使用する指示があった場合のフローチャート。

【図 5】

実施例 1 の目的プログラムファイルの構造図。

【図 6】

本発明のコンパイラの実施例 2 の構成図。

【図 7】

実施例 2 のソースプログラム解析部のフローチャート。

【図 8】

実施例 2 の目的プログラムファイル作成部のフローチャート。

【図 9】

実施例 2 のコンパイラへのソース比較指示をおこなう場合のフローチャート。

【図 1 0】

実施例 2 のコンパイラへのソース出力指示をおこなう場合のフローチャート。

【図 1 1】

実施例 2 の目的プログラムファイルの構造図。

【図 1 2】

実施例 2 のソースプログラムファイル例。

【図 1 3】

実施例 2 の目的プログラムファイルの構成例。

【図 1 4】

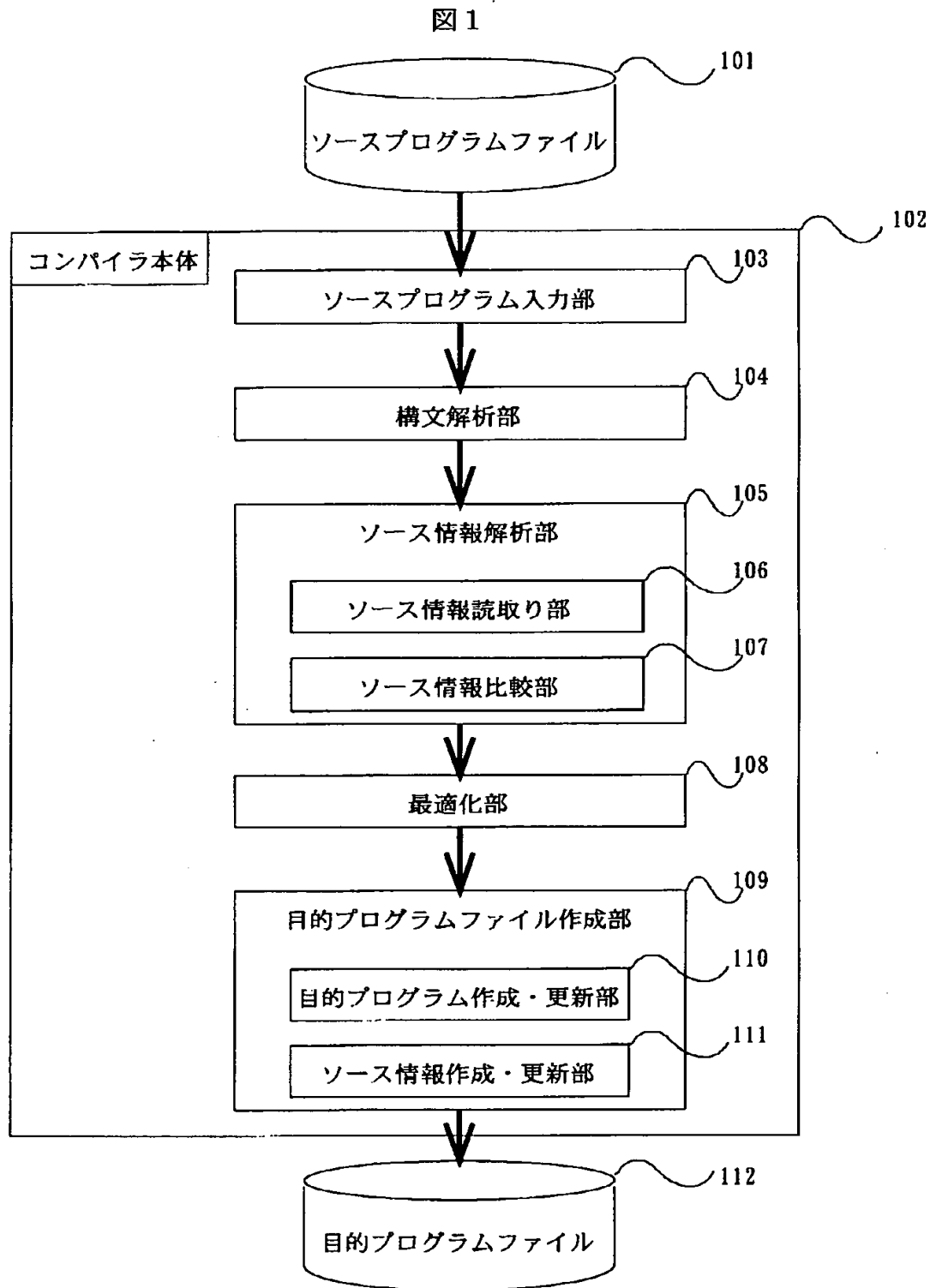
実施例 2 の別の目的プログラムファイルの構成例。

【符号の説明】

101…ソースプログラムファイル、102…コンパイラ本体、
103…ソースプログラム入力部、104…構文解析部、
105…ソース情報解析部、106…ソース情報読取り部、
107…ソース情報比較部、108…最適化部、
109…目的プログラムファイル作成部、110…目的プログラム作成・更新部、
111…ソース情報作成・更新部、112…目的プログラムファイル、
201～205…ソース情報解析部の処理手順、
301～305…ソース情報作成・更新部の処理手順、
401～402…利用者からコンパイラへの指示があった場合処理手順、
501～502…目的プログラムファイルを構成する情報、
601…目的プログラムファイル認識部、602…ソースプログラム読み取り部、
603…ソースプログラム比較部、604…最適化処理部振り分け部、
605…目的プログラム生成部、606…目的プログラム出力部、
607…ソースプログラム出力部、

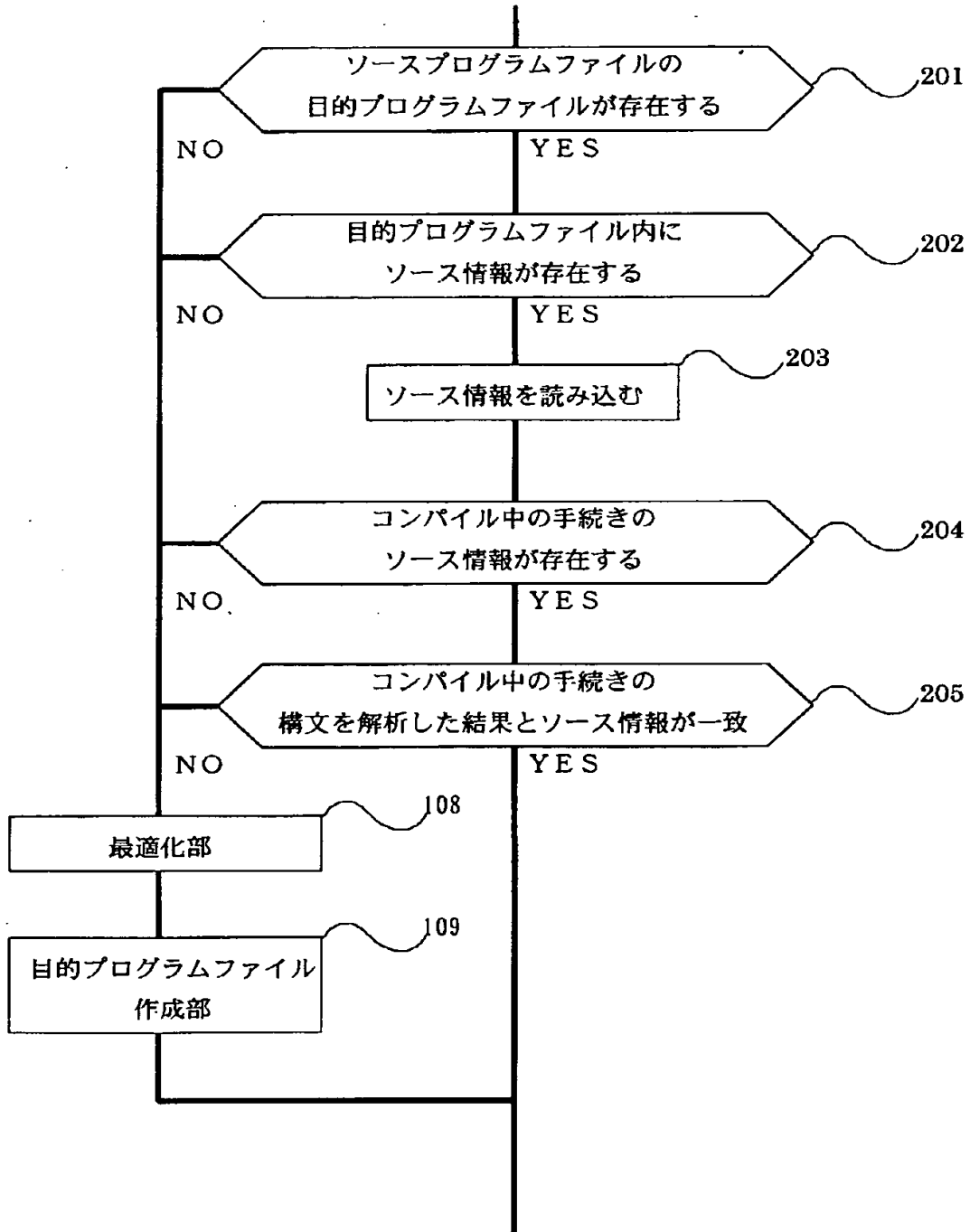
【書類名】 図面

【図1】



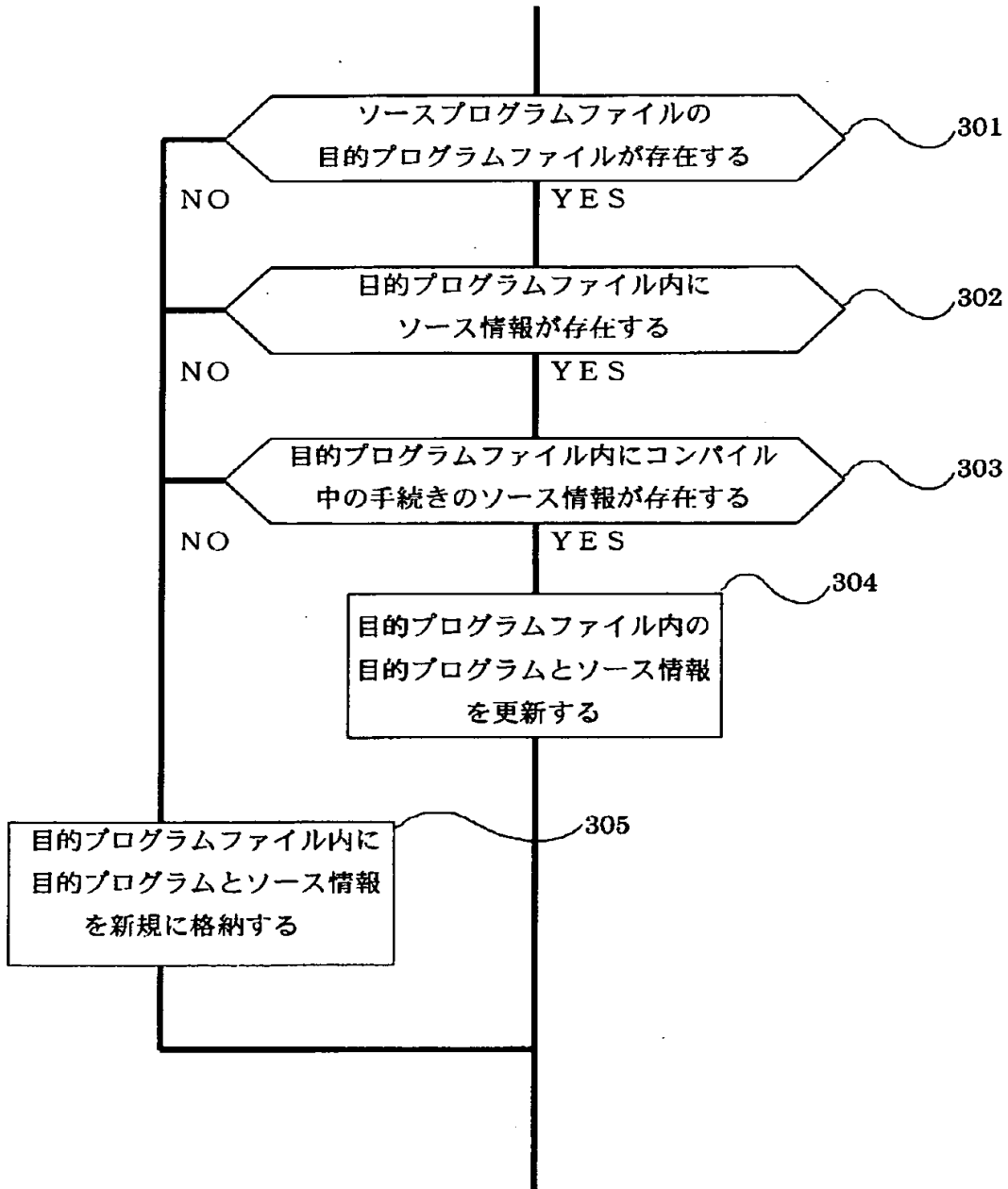
【図 2】

図 2



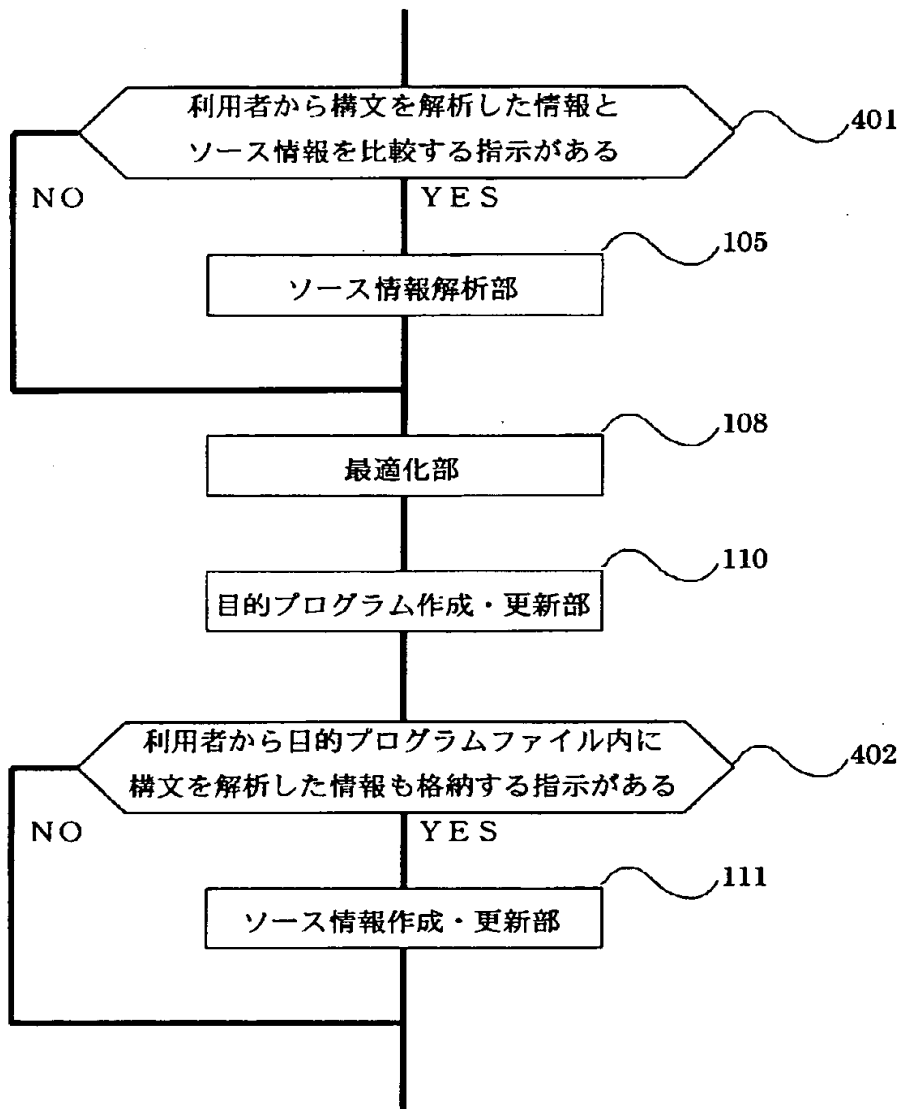
【図 3】

図 3



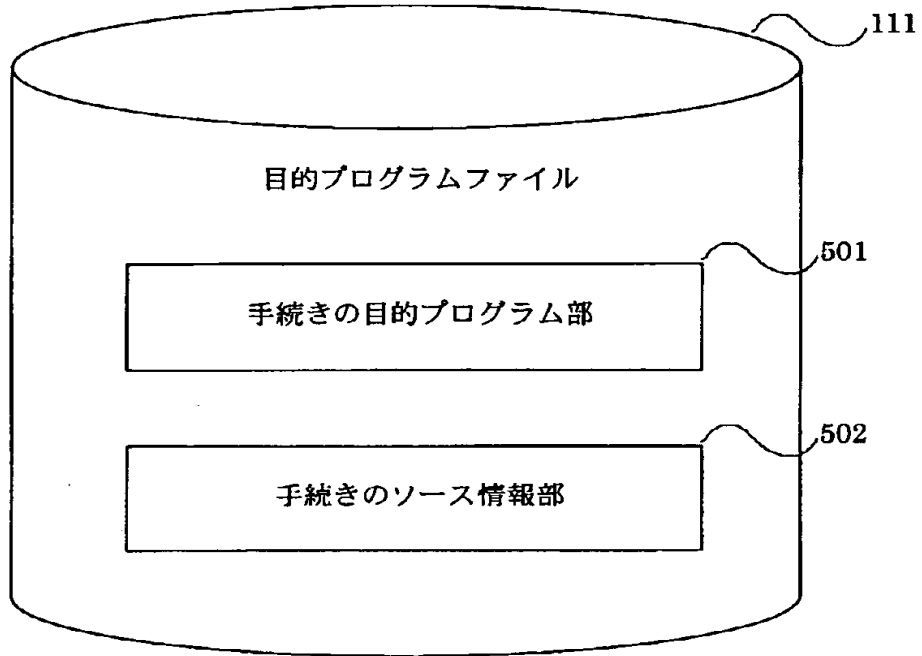
【図 4】

図 4



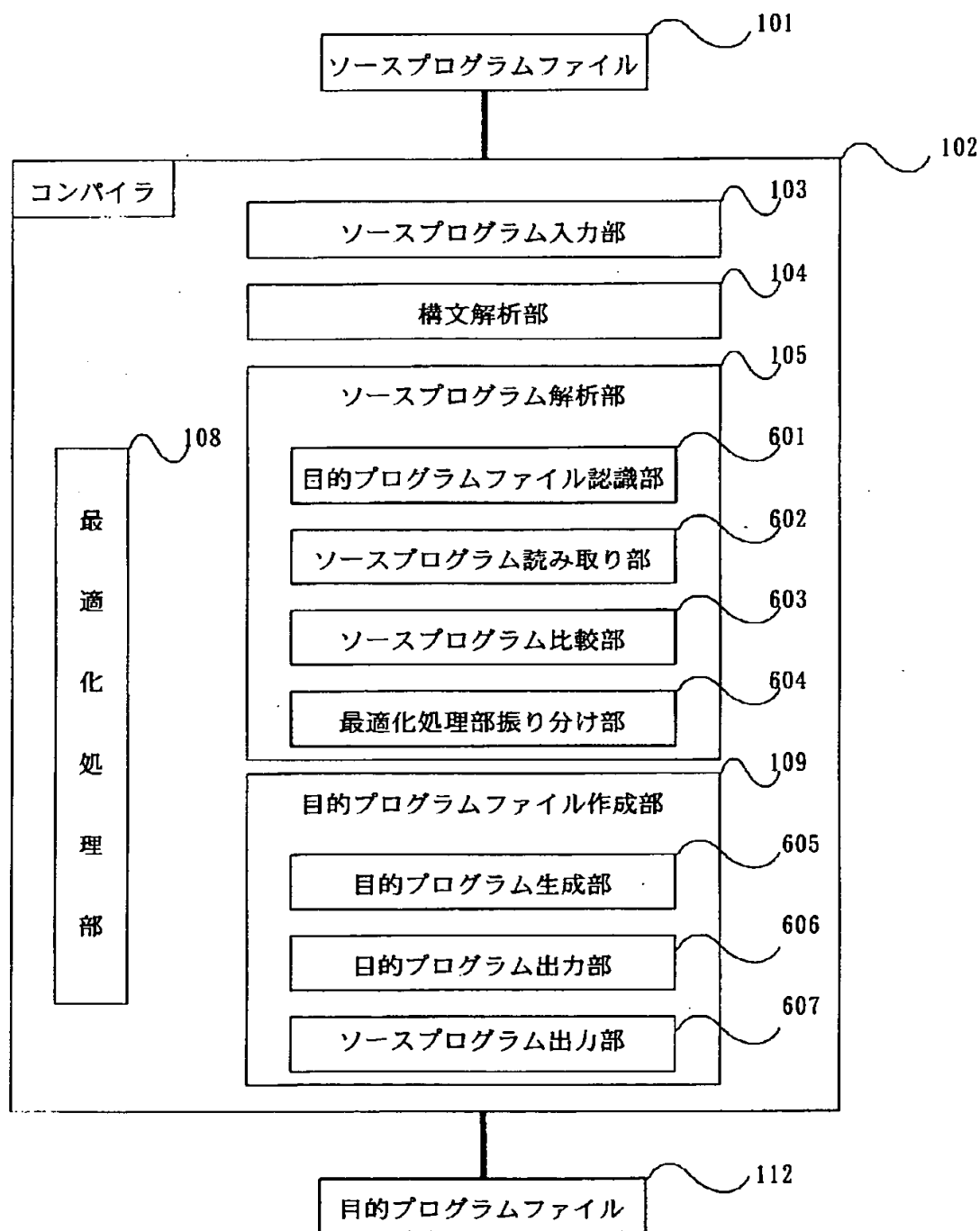
【図 5】

図 5



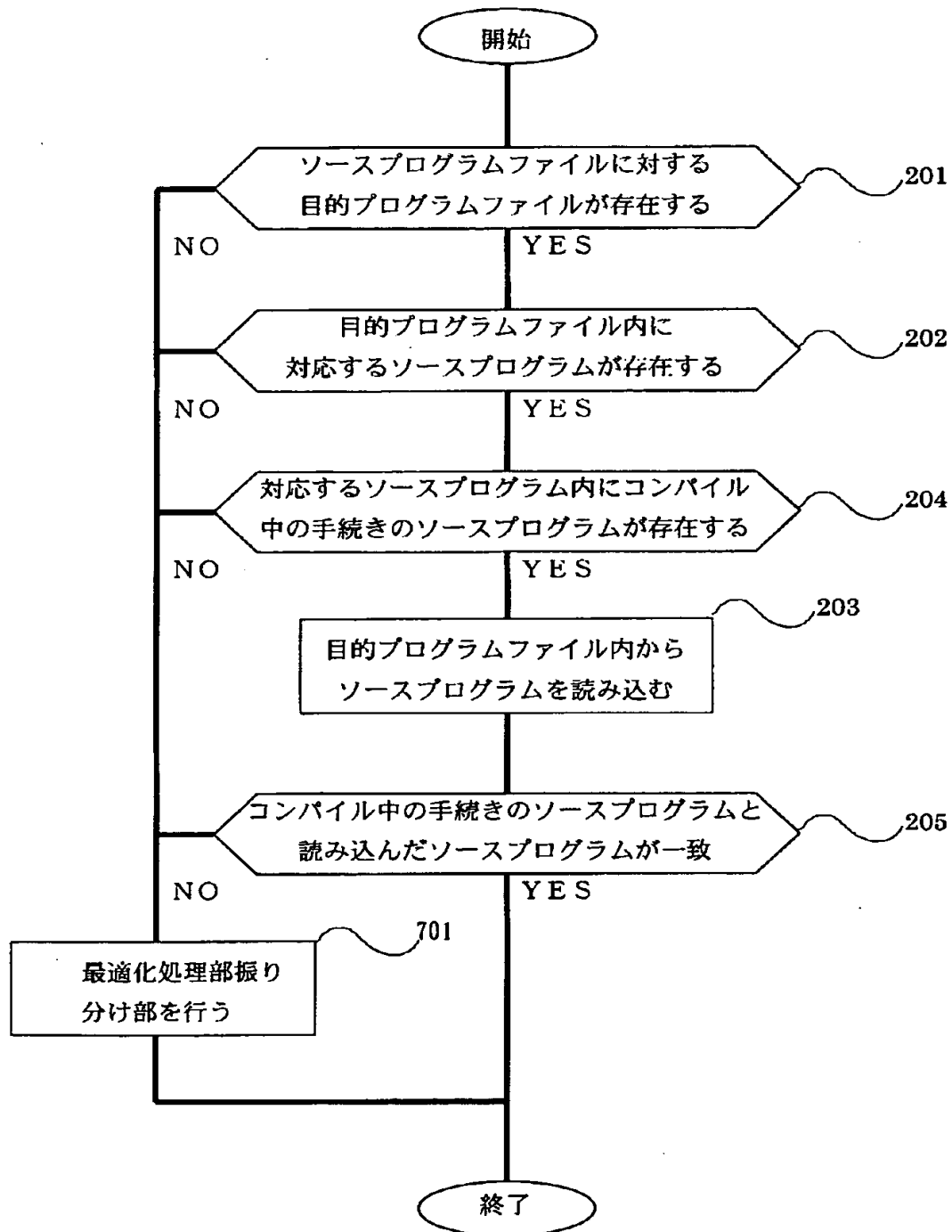
【図 6】

図 6



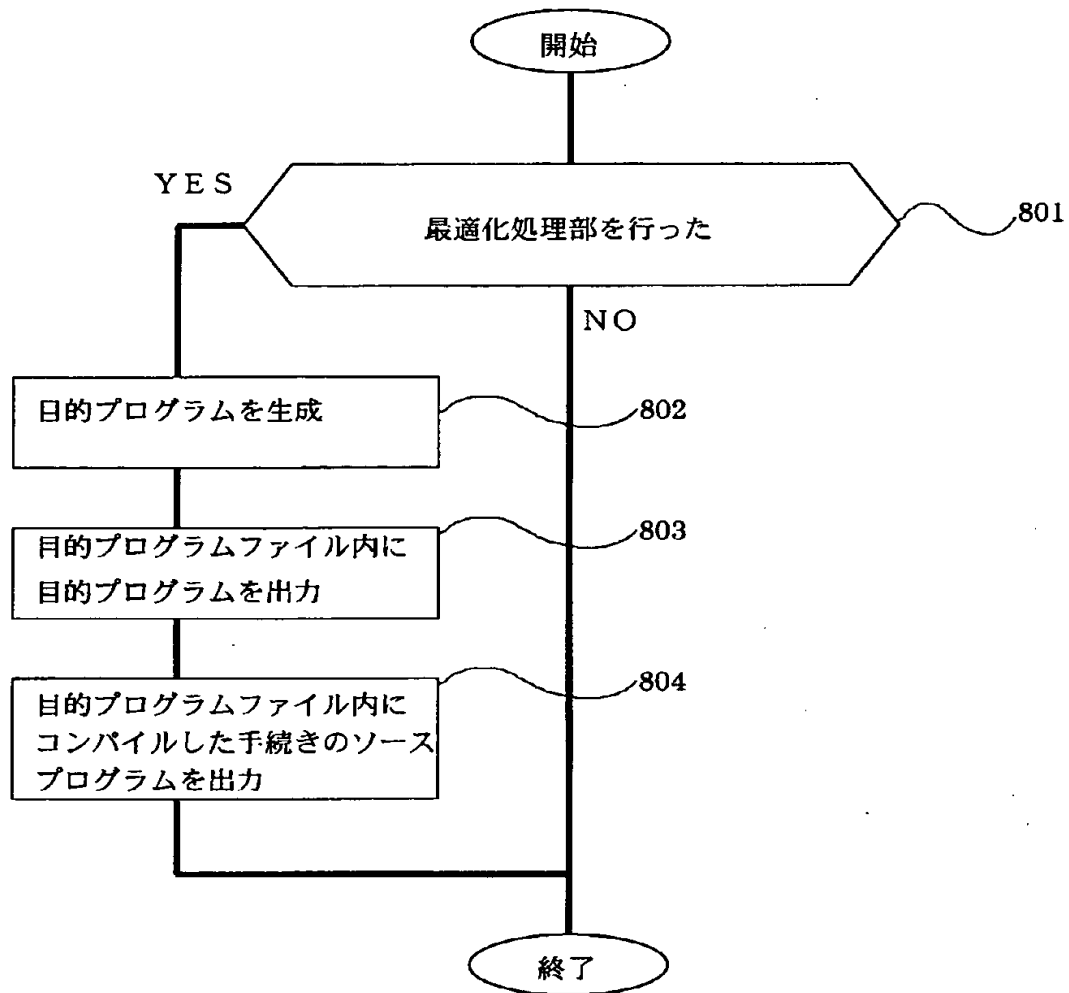
【図 7】

図 7

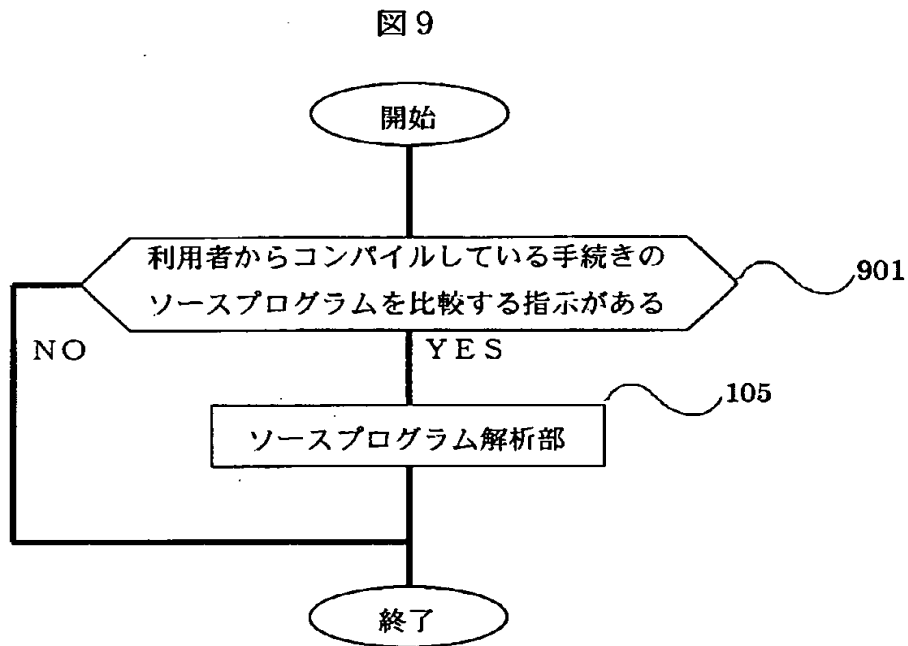


【図 8】

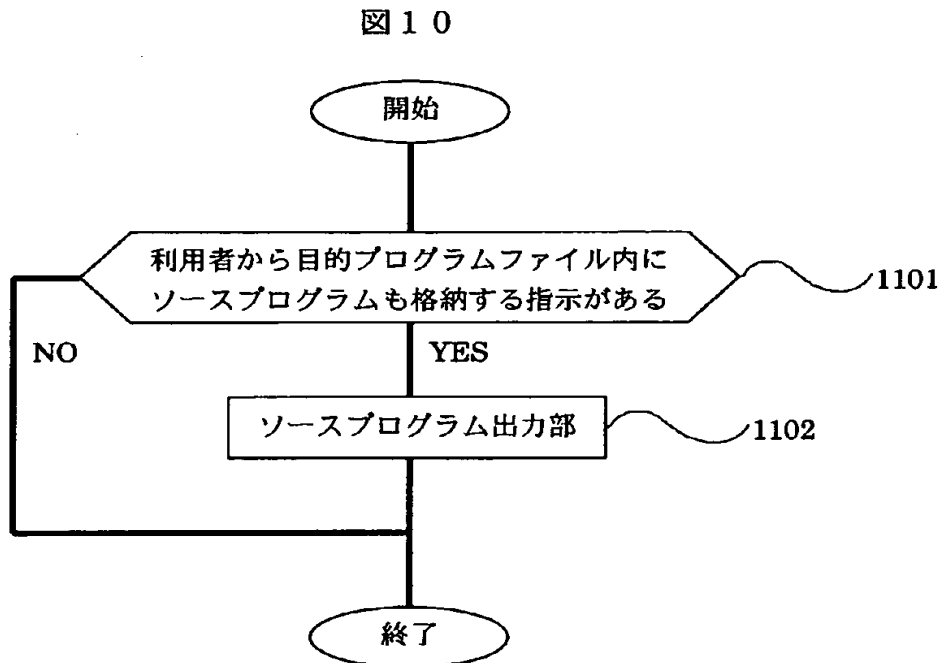
図 8



【図 9】

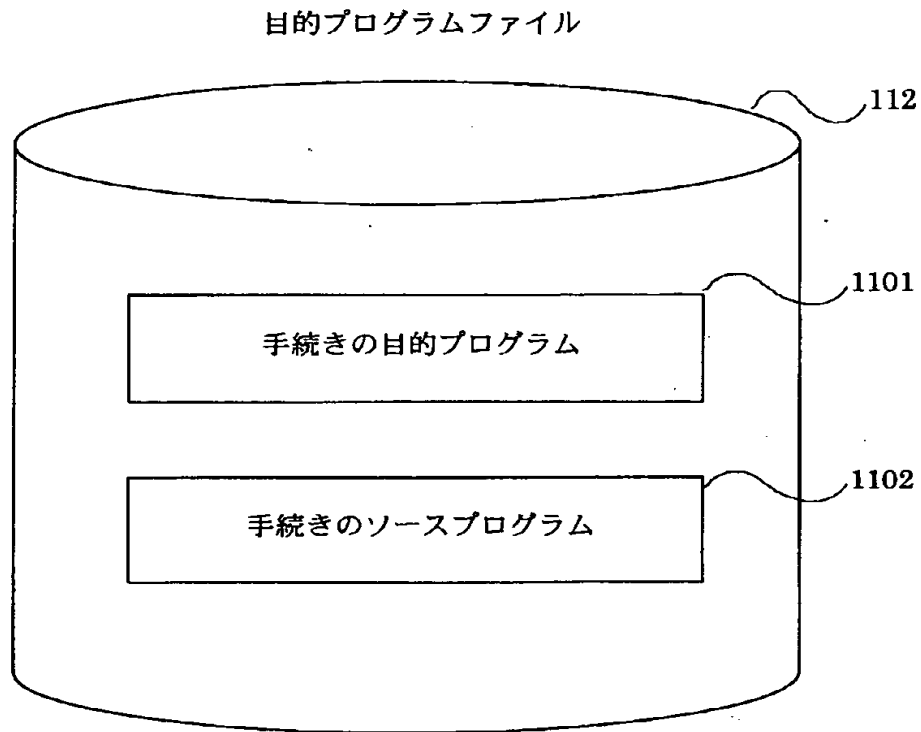


【図 1 0】



【図 1 1】

図 1 1



【図 1 2】

図 1 2

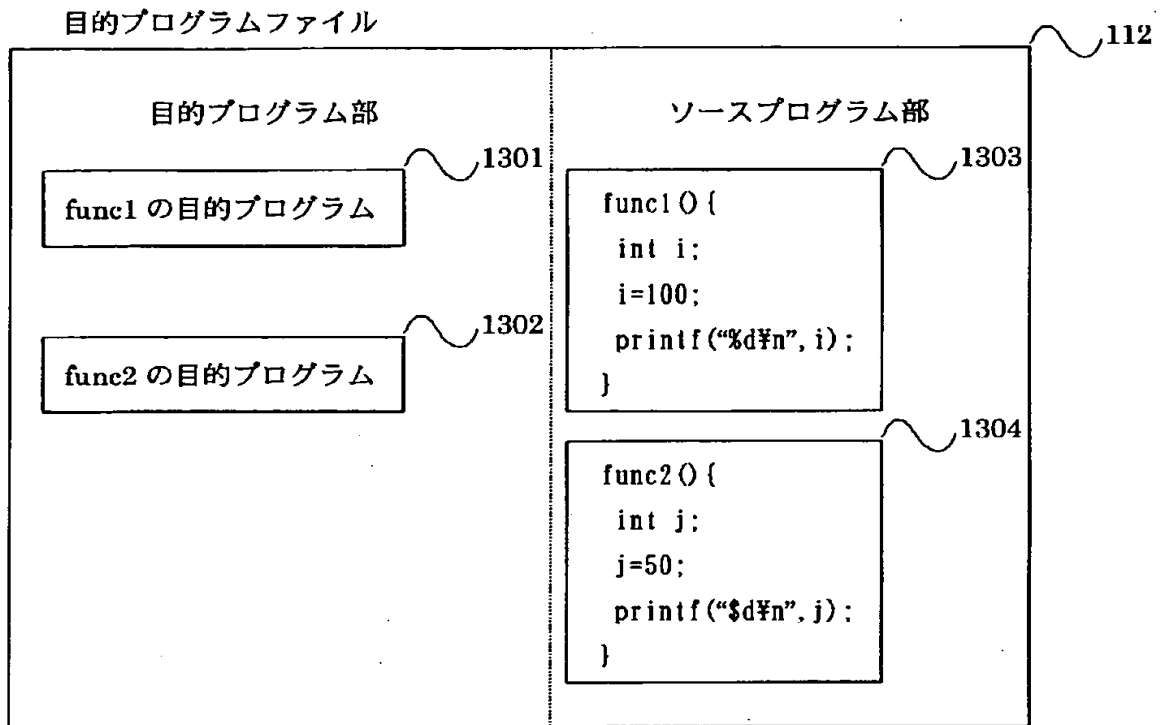
ソースプログラムファイル

The diagram shows a rectangular box representing a 'ソースプログラムファイル' (Source Program File). The box is pointed to by reference numeral 101. Inside the box, the following C code is written:

```
func1(){  
    int i;  
    i=100;  
    printf("%d\n", i);  
}  
func2(){  
    int j;  
    j=50;  
    printf("%d\n", j);  
}
```

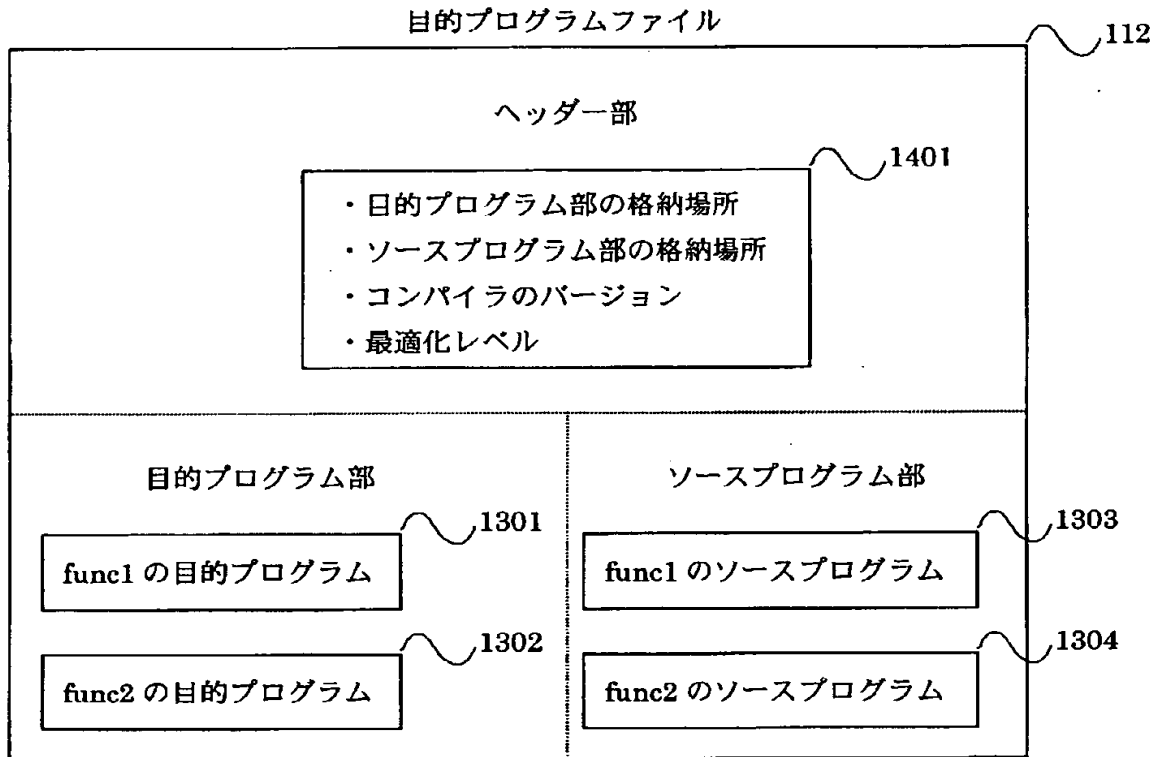
【図 13】

図 13



【図 1 4】

図 1 4



【書類名】 要約書

【要約】

【課題】

複数の手続きが記述されたソースプログラムファイルをコンパイルした後、そのソースプログラムファイル内の1つの手続きを変更して再びコンパイルすると、変更していない手続きを含めた全ての手続きについて再びコンパイルする。その為、わずかな変更でもコンパイル時間が必要になる問題がある。

【解決手段】

目的プログラムファイル作成部 1 0 9 で、目的プログラムファイル 1 1 2 内に、コンパイルした各手続きのソースプログラムを格納しておき、2回目以降のコンパイル時に、ソースプログラム解析部 1 1 5 で、格納した各手続きのソースプログラムと、コンパイルしている手続きのソースプログラムを比較し、変更された手続きを検出する。手続きが変更された場合のみ、以後のコンパイル処理を行い、手続きが変更されていない場合は以後のコンパイル処理を省略する。

【選択図】 図 6

認定・付加情報

特許出願の番号	特願 2000-332110
受付番号	50005049829
書類名	特許願
担当官	濱谷 よし子 1614
作成日	平成 12 年 10 月 31 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	000005108
【住所又は居所】	東京都千代田区神田駿河台四丁目 6 番地
【氏名又は名称】	株式会社日立製作所

【特許出願人】

【識別番号】	000233055
【住所又は居所】	神奈川県横浜市中区尾上町 6 丁目 8 1 番地
【氏名又は名称】	日立ソフトウェアエンジニアリング株式会社

【代理人】

申請人	
【識別番号】	100075096
【住所又は居所】	東京都千代田区丸の内 1-5-1 株式会社日立 製作所 知的所有権本部内
【氏名又は名称】	作田 康夫

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日	1990年 8月31日
[変更理由]	新規登録
住 所	東京都千代田区神田駿河台4丁目6番地
氏 名	株式会社日立製作所

出 願 人 履 歴 情 報

識別番号 [0 0 0 2 3 3 0 5 5]

1. 変更年月日	1 9 9 0 年 8 月 7 日
[変更理由]	新規登録
住 所	神奈川県横浜市中区尾上町 6 丁目 8 1 番地
氏 名	日立ソフトウェアエンジニアリング株式会社